

Vex™ for the Technically Challenged

(by the Technically Challenged)

Version 3.1

© Yolande Petersen and Justin Petersen 2006, 2007, 2008

Note: This material may be distributed free of charge for educational/non-profit purposes. It is "copyrighted" in the sense that we would appreciate acknowledgment if significant portions are copied, and we would be royally ticked if we found this volume for sale under someone else's name.

Table of Contents

Foreword

1. Getting Started
2. Locomotion (Getting Around)
3. Manipulation of Objects (Lifting and Gripping)
4. Programming
5. Sensors
6. Transmitter Considerations
7. Simple Challenges
8. Further Exploration
9. References

Appendix A: Recommendations for Add-ons

Appendix B: Cutting Suggestions for Commonly Used Pieces

Foreword

To date, relatively little documentation is available on the Vex™ robotics system, and much of what is available requires a prior level of robotics experience. This guide is intended for those who are starting with Vex™ from ground zero or are moving up from a simpler system like LEGO™ Mindstorms.

You may find that much of the information here is painfully simplistic and obvious. It was collected as the result of many "duh" moments, the culmination of much time ~~wasted~~ invested over the past 3 years. We began our Vex™ journey in 2005 shortly after the release of the Vex™ platform but as a team with a non-engineering coach and largely self-taught students, we found it difficult to advance, due to the lack of training materials. Cool designs on the Innovation First, Inc. website¹ and Chief Delphi forum² provided tantalizing building ideas, but we were unable to reverse engineer many of these designs without an overall understanding of basic engineering principles. With no nearby mentorship, we were left to scouring the internet for assistance and ideas. It was our hope that over time, 2 types of resources would emerge:

1. An overview text explaining basic engineering principles as they apply to Vex™, and
2. A builder's guide of projects, with step-by-step instructions for a few cool projects.

To our knowledge, neither of these resources has yet materialized in published form, and we have often lamented that SOMEONE should develop these projects. And as we have laboriously inched forward, we have fewer reasons to exempt ourselves from being that someone. This tome is an attempt to fill the need for the first type of resource, and *Vex™ Machinations: A Step-by-Step Project Guide* is designed to fill the second. In many cases, what we present is not an optimal or even a good solution, but simply something that gets the job done, by hook or by crook, leaving the best challenges for you to solve. We know that there are more knowledgeable builders out there, and it is our hope that they will provide better resources where ours are lacking.

We would like to thank Art Dutra IV, Kathie Kentfield, Blake Ross, and the Chief Delphi forum community for patiently answering our obvious questions, Team Metal Gear for continuing to add to their and our catalog of knowledge, Keith Petersen for editing assistance, and Jim and Conni Bock for their instruction and inspiration. We have attempted to acknowledge all sources and photos, but if we inadvertently missed someone, please let us know. Unlabeled photos were taken from our personal library.

Yolande Petersen, Team Mom & Coach
Justin Petersen, Technical Guy

Getting Started

Storage

After you unpack your Vex™ kit, it is advisable to purchase a storage container with compartments, like a tackle box, for the various pieces. Ideally, the compartments should be of varying sizes: some long enough for the long bars and axles, and some small enough for the screws, nuts and washers. The cardboard box which the kit comes in is useful for storing larger items like the remote, wheels, and the manual.



Photo used by permission from Art Dutra

Manual (the Inventor's Guide)

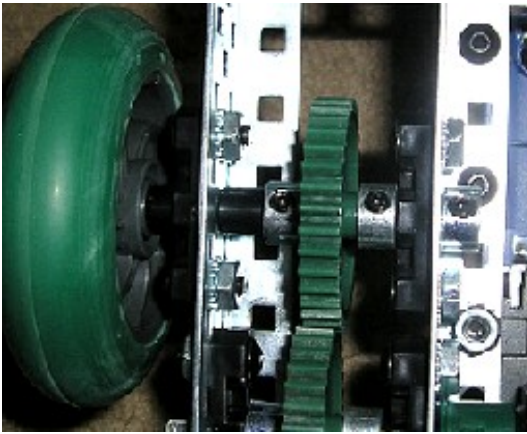
The Inventor's Guide should be filled with pages and separated into sections. Some of the sections are empty and will be filled up as you purchase additional accessories and add the documentation to your manual. It's a good idea to read through the manual, even though "nobody ever reads the manual." If your kit came without an Inventor's Guide, it can be downloaded here:

<http://www.vexrobotics.com/docs/inventors-guide/main/vex-inventers-guide-9-27-06.pdf>

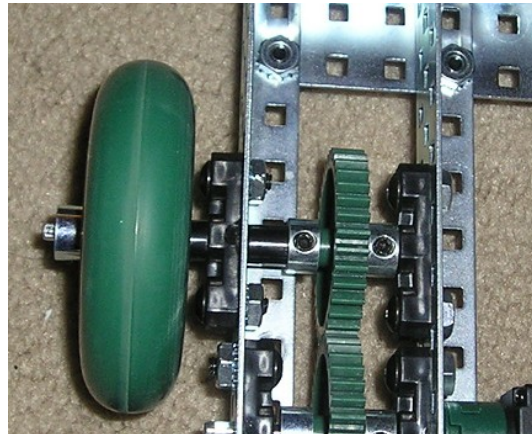
Squarebot

From the Inventor's Guide, you can build the Squarebot (instructions beginning in the Structure section, 2.2-2.26). This will help you become accustomed to the parts and tools in the set, as well as giving you a robot that actually works. Try driving the robot using the transmitter (aka remote control) on a variety of terrain to test the limits of your robot's speed and power.

One design improvement is suggested here. Notice that the Squarebot wheels are held onto the axles by friction, and that they tend to fall off. This problem becomes exacerbated with repeated use. Securing a collar to the outside of the axle solves the problem; however, if the Squarebot is built as shown, there is insufficient axle for this purpose. Removing the innermost collar allows the axle to be slid outward, and proper positioning of the remaining collars against the rails prevents sliding of the axle back and forth. Also, on p. 2.20, the use of spacers is suggested if the gears slide; these are generally required.



Original Squarebot axle, collar on inside



Modified Squarebot axle, collar on outside

Modification of Parts

One thing which distinguishes the Vex™ system from many other robot kits is that its pieces are meant to be modified through bending and cutting. While it is wise to be conservative about cutting, a balance needs to be found between settling for an inferior design to avoid cutting, and wasting materials. Keep all cut pieces, no matter how small! They will almost always be reused in some way.

Having some pieces pre-cut will make your project go faster, an important consideration if you have a team of impatient students. There are some sizes of pieces that will likely be used if you use your Vex kit extensively. If you decide to upgrade by adding a few parts (see Appendix A), there are recommendations for cutting (see Appendix B) which will give your kit maximum flexibility.

Minimal cutting tools required include tin snips, a hacksaw for angle bars and square bars (including a vise for securing pieces), and a file for removing burrs. Alternatively, bolt cutters can be used for square bars, with varying results.

Another helpful item is a small platform for elevating the center of your robot, keeping the wheels off the ground. This enables you to download programs or perform tests with the wheels spinning without having the robot run around the room, tangling the cords. A stack of paperback books or 2" X 4" blocks does the job.

Upgrades and Add-ons

While it's sad to think that you can barely get your kit unpacked before thinking about spending more money, there are a few additional purchases you should consider that will add to the enjoyment of your kit. What you purchase will depend on what you plan to do with your kit. If your main focus is on building, you will want to add to your collection of parts, especially to replace the ones you ~~mutilate~~ modify. One way to do this is by purchasing the Vex™ Metal and Hardware Kit. However, over the past 3 years, we have found that certain parts are commonly used, and we have compiled a list of what we consider essential add-ons, for about \$100 (see The Bang for Buck Upgrades - Appendix A). It's our (humble) opinion that the purchase of these parts provides the greatest flexibility for a relatively small expenditure.

Another popular item is the Programming Kit, which comes in 3 flavors: EasyC, ROBOTC, and MPLab. EasyC is icon-based, making it a good language for first-time programmers. ROBOTC is text-based, but menu-driven. It has the advantage of being available in versions that are compatible with LEGO™ Mindstorms NXT, and a number of other hardware systems. Thus, ROBOTC is a versatile language if you plan to migrate across platforms. MPLAB is generally preferred by advanced users. For a more detailed comparison, see: <http://www.vexrobotics.com/vex-robotics-programming-kit.shtml>.

Another almost universally used item is the Vex™ Power Pack, which contains rechargeable batteries that will save money over the long haul. The 7.2 battery can also be replaced individually when it finally bites the dust. Also, some competitions require use of the Power Pack (rather than individual AA batteries).

If you are planning to purchase a kit for a competitive team, you might find the Vex Classroom Lab Kit to be a good value. It will need the addition of a Programming Kit, but otherwise is a "no brainer" way to accumulate a good supply of useful parts needed for a reasonably competitive robot.

Additional add-ons will depend primarily on your wants and your wallet. So many parts, so little money! Appendix A has more recommendations.

Vex™ accessory packs often have "goodies" hidden in the packaging. Be sure to completely empty the box and check for documentation. Unfold/unwind all cardboard packaging, and cut or pull off any large pieces of packing tape to be sure that you are not discarding small hidden treasures. Better yet, perform an inventory of all the parts listed on the side of the box. Most accessory packs come with documentation that is intended to be stored in the Inventor's Guide binder.

Locomotion (Getting Around)

Gears, Speed, and Strength

You may have observed that your Squarebot is relatively fast and can navigate fairly rough terrain. In case you're wondering, "Can I make my robot go faster?" and "Can I make it stronger?" the answer is "yes" to both, but you can't do both at the same time.

Take a look at the gear train on your Squarebot. The motor is connected to a 60-tooth gear, while the wheel axles are connected to 36-tooth gears. Each time the motor makes one complete rotation, it moves the large gear through 60 teeth, which in turn, also moves the smaller gear through 60 teeth. However, 60 teeth on the 36-tooth gear represent nearly 2 rotations (one and $24/36$ rotations, to be precise), which means that the wheels have a faster turning speed, or *angular velocity*, than the motor. You have just found a way to make your robot go faster! This process of powering a larger gear to make a smaller gear go faster is called *gearing up* (think of a race car).

While increasing speed can be a good thing, the trade-off is a reduction in strength, or *torque*, and at times, it is desirable to reduce speed and increase torque by *gearing down* (think of a tank). This is accomplished by powering a small gear and using it to drive a larger gear.

Ground speed, or *linear velocity*, can also be modified by changing the wheel size. If the motors have a fixed number of rotations per minute, using larger wheels (with a larger circumference) will enable the robot to cover more distance with each rotation.

Calculating the gear ratio (drive: driven) can give a precise calculation of how much speed (or torque) is gained. In the case of the Squarebot, the ratio is 60:36, or reduced, 5:3. We can obtain a ratio of 5:1 by using the 60-tooth gear to drive the 12-tooth gear, and even higher ratios can be obtained by using more than 2 gears in a gear train. One caution is that some torque is always lost to friction with each conversion, so minimizing the number of gears in a gear train is generally desirable (even though the longer trains look more impressive). More on gearing can be found in the Structure subsection of the Inventor's Guide.

Driving Base Considerations³

The driving base is the foundation of all other aspects of robot design. Optimally, you want to find the right balance of speed, strength, stability, and maneuverability to accomplish the desired task. Obviously, the optimal base needed for driving at maximum speed on a smooth, flat surface will differ from one needed to climb up a 20-ft. garbage pile. As described above, the balance between speed and strength is accomplished by adjusting the gear ratios between the input (motors) and the output (wheels).

Maneuverability can be defined as the speed and precision with which the robot can change direction. While a high degree of maneuverability can be desirable, it can actually be a disadvantage at times, as a robot which changes direction too easily can be difficult to control.

Two-Wheel Drive

The *two-wheel drive*, or *castorbot*, is highly maneuverable and relatively easy to build. It requires only two drive wheels (one on each side) and support of the chassis to

keep the robot from tipping over. Supporting the chassis with two free-spinning wheels is possible, but the friction on the unpowered wheels often makes turning difficult. Removing the rubber from the "dead" wheels greatly reduces the friction. Alternatively, a castor may be used to support the chassis. However, a traditional pivoting castor (like on a swivel chair) is often unpredictable, as its starting position affects the direction of motion. A better choice is an omni wheel or a skid plate (a low-friction object which simply drags on the ground). Creating a skid plate from an object with a rounded surface, like a rubber-less wheel or the Vex™ bearing block, is preferable to objects with corners, which snag easily.

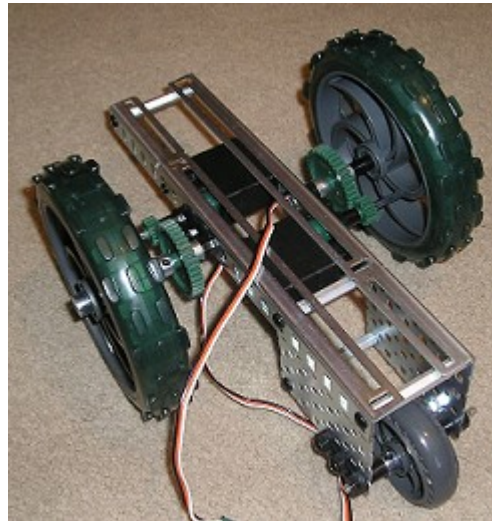
There are several disadvantages to the castorbot. It turns with relative ease because there is virtually no side friction working against the wheels of the robot as it turns. However, with no friction regulating the robot's speed of turning, its inertia will always make it "want" to continue turning even after the motors have stopped. In addition, the unpowered wheels or castors have no pushing power. Finally, castorbots are often difficult to drive in reverse.

There are ways to offset or compensate for these disadvantages. When designing a two-wheel drive train, the powered wheels should be close to the center of gravity. This allows the robot's pivot point to remain close to its center of mass, minimizing the area through which it must travel in order to turn.

Below are 2 configurations of a simple castorbot (controller & battery pack removed for better view of gearing). It is geared down to compensate for the large wheels and turns easily. However, one design flaw should be noted, the result of a tightwaddish reluctance to cut 2 additional rails. Note that the axles are cantilevered; that is, they are supported on only one side of the gearing. This is an unstable design because plastic bearing blocks are bendable and do not perfectly align with the metal. If too much weight is applied to the center, the axles bend inward, producing wobbly motion. A better design would be to insert another set of rails between the gears and the wheels, supporting the gear train on both sides.



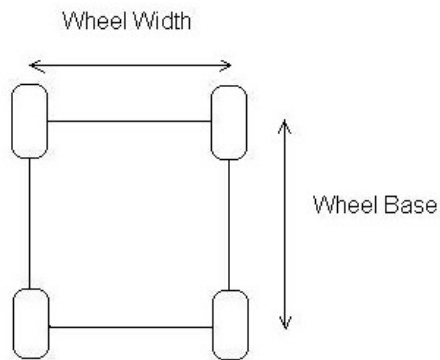
Drive wheels far from the center of gravity (CG), with significant drag from the castor.



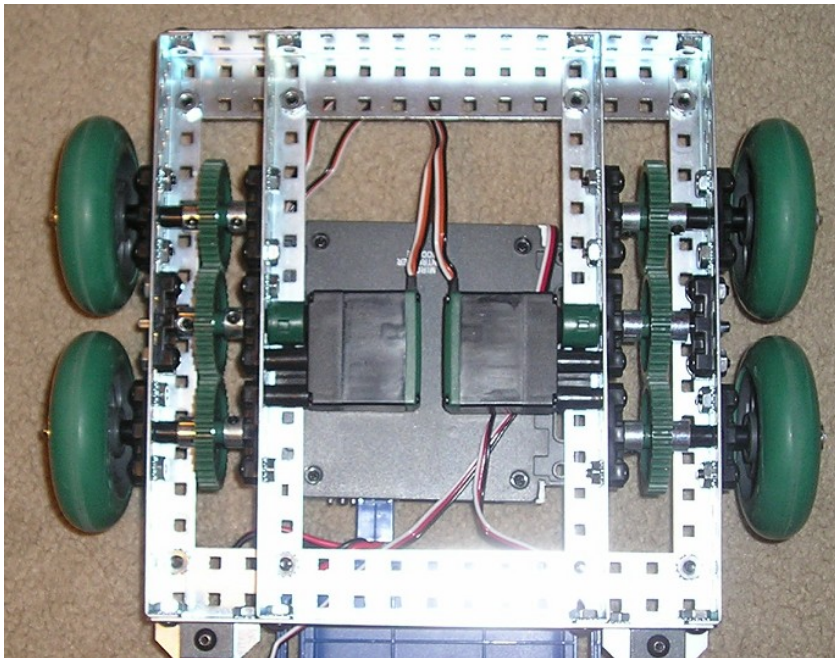
Drive wheels closer to CG, less drag.

Four-Wheel Drive

Another type of drive train is the *four-wheel-drive* system; the Squarebot is an example of this. With the addition of two extra driven wheels, a four-wheel-drive robot has more traction and control than a two-wheel-drive robot. The trade-off is the increased wheel base (distance from front to back wheels), which can cause problems turning, especially when the wheel base exceeds the wheel width (distance from left-side to right-side wheels). On the other hand, the longer wheel base adds stability – the robot is less likely to tip when stopping suddenly. It is thus more capable than a castorbot of sporting large attachments, especially ones with a high center of gravity.

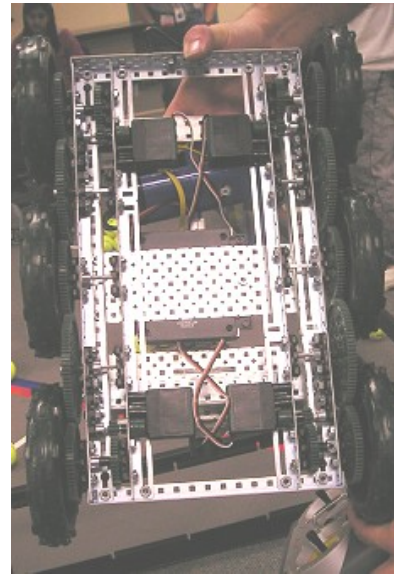
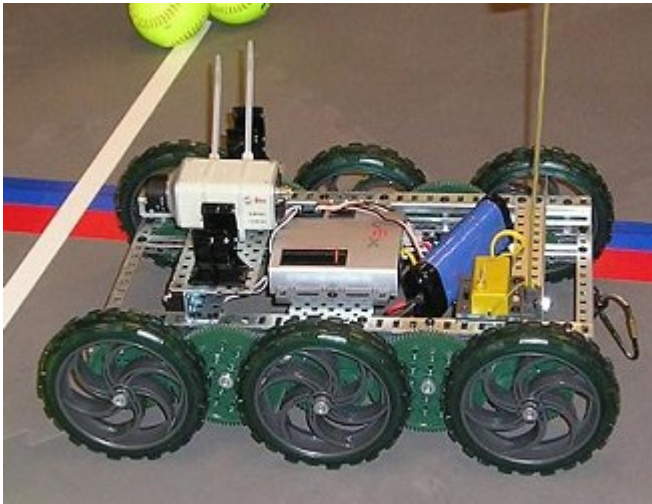


Another 4-wheel drive example is the Slowerbot (pictured below, upside down), geared with a 1:1 ratio.



Six-Wheel Drive

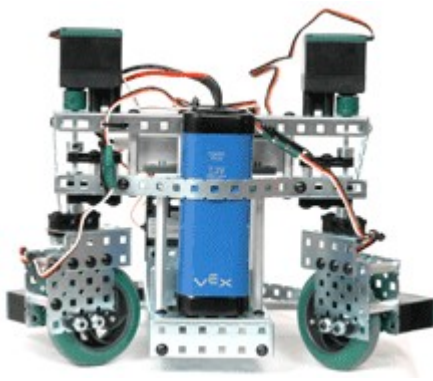
The *six-wheel-drive* robot offers a good compromise between traction and maneuverability. Commonly, the center wheels are approximately 1/8" - 3/16" lower than the front and back wheels. Thus, the robot rests on only 4 of its 6 wheels at any given time, so the effective wheel base is only the distance from the center wheels to the front or back, or about half the distance from front wheels to back wheels. This shortened base allows the robot to turn more easily, while greater total chassis length provides more stability.



Photos used by permission from Mark Edelman

Swerve Drive

A *swerve drive* (aka crab drive) robot is able to rotate its wheels, resulting in a great increase in maneuverability. The disadvantage is that this rotation mechanism is more complex to build, and the pivoting wheels make the robot easier for other robots to push around. An example is the Swervebot, built by Chris Carnevale & John V-Neun, which can be found at vexlabs.com/vex-triplets.shtml (with more photos and video of the Swervebot available). Vexlabs also sells a Swerve Drive Kit which can be used as an add-on to the Starter Kit.



Photos used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI.

Holonomic Drive

Vex™ omni wheels (purchased separately) make it possible to construct a *holonomic drive*, or *omni drive*. A holonomic drive enables a robot to rotate in place, move in any direction, or do both at the same time. The major advantage is a great increase in maneuverability without having to add an entirely new mechanism to turn the wheels – a holonomic drive can change direction without changing wheel position. The major disadvantage is that the robot isn't very good when it comes to pushing and shoving. Omni wheels tend to have poor traction, as they are generally made from hard, low-friction material without treads. Also, the wheels are controlled independently, adding complexity.

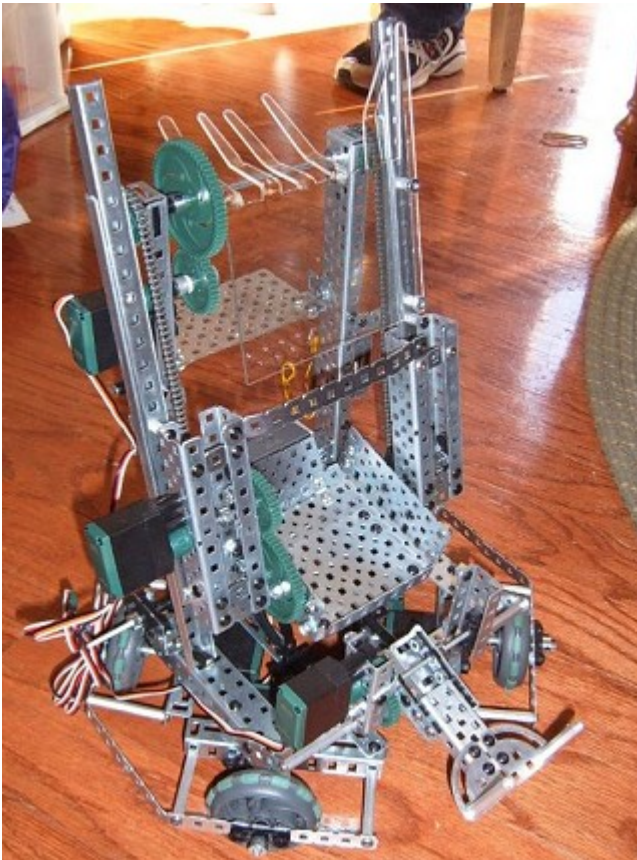
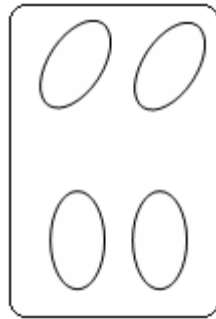


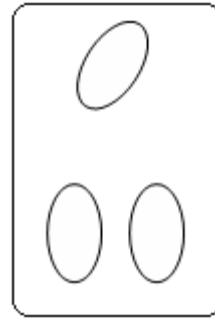
Photo used by permission from Art Dutra

Steering Drives

Many simple robots are constructed with one motor on each side, using the difference in speed for turning (these are called *differential drives*). Thus, the same motors are used for controlling speed and direction. However, when you think of how a "real" car is driven, only one control (the gas pedal) is used to drive forward, and a separate control (the steering wheel) is used for steering. A robotics car-type drive (whose steering mechanism turns the 2 front wheels) and a tricycle drive (with a 1-wheel steering mechanism) use this principle.



Car-type Drive

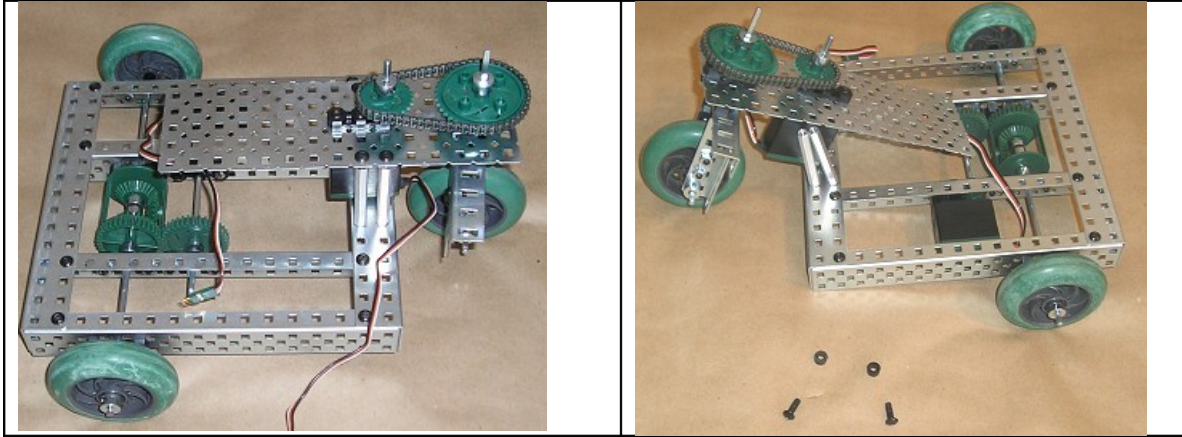


Tricycle Drive

In order for a car to turn, the inside and outside wheels must travel at different speeds. With only one mechanism controlling forward motion, how is this possible? One way is to use a *differential* or *differential gear* (not to be confused with a differential drive which uses 2 motors). A differential distributes unequal amounts of "spin" to the 2 axles on either side. The Wikipedia entry shown below nicely describes how this works.

<p>Input torque is applied to the ring gear (blue), which turns the entire carrier (blue), providing torque to both side gears (red and yellow), which in turn may drive the left and right wheels. If the resistance at both wheels is equal, the planet gear (green) does not rotate, and both wheels turn at the same rate.⁴</p>	<p>If the left side gear (red) encounters resistance, the planet gear (green) rotates about the left side gear, in turn applying extra rotation to the right side gear (yellow).¹</p>

Below is a tricycle drive robot. Note that a single motor drives the rear wheels through the differential, while the second motor (a servo) is used for steering.



Chassis Layout

The geometry of the chassis is also an important consideration, and a number of configurations are pictured in the 1st row of the diagram below, taken from Team Unlimited's PowerPoint⁵. Chassis are commonly rectangular, and the ratio of wheel base to wheel width can be adjusted. A short wheel base & wide body (pictured in the upper left) are better for turns and side stability, while a long wheelbase & narrow body (2nd from left) are better for front-to-back stability. A square chassis is a compromise for both. U-shaped or H-shaped chassis are more fragile than rectangular chassis, but are useful when trying to stay within a size limit while making room for a manipulator which touches the ground, as demonstrated by the 3rd object in the 2nd row of the diagram. While the wheel axles are typically attached to the chassis (1st object, 2nd row), the chassis may ride above the wheels (2nd object, 2nd row).

Sample Vex chassis configurations

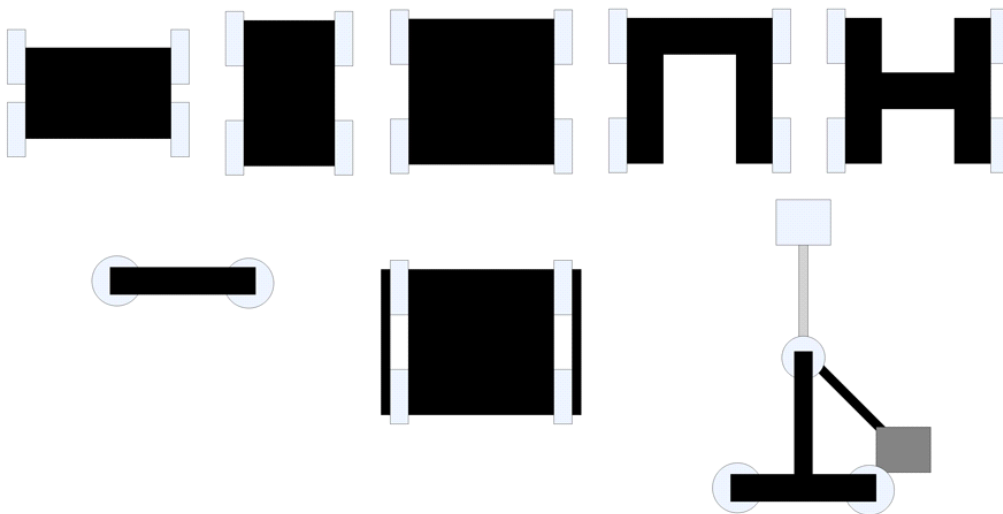


Diagram used by permission from FVC 2013 ('06-'07) Team Unlimited.

Pictured below is a U-shaped chassis. Note that the absence of a bar across the front allows ample room for the ball collector.



Scorpio; photo used by permission from FVC 3220 ('06-'07)
S.P.A.M.

Manipulation of Objects (Gripping and Lifting)⁶

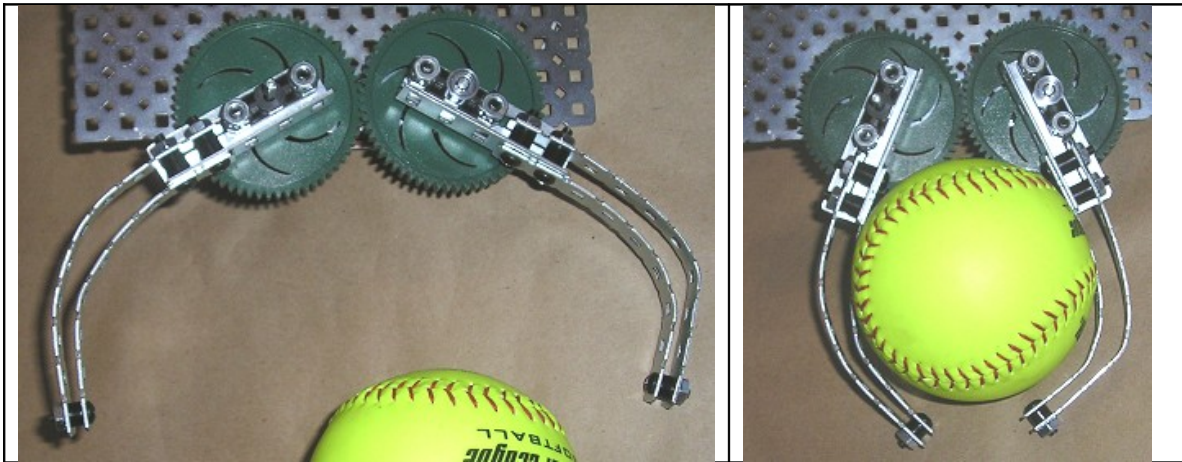
One of the important applications of robotics is the ability to move objects. Handling dangerous objects (like land mines) and carrying heavy objects (like food supplies and homework) are just a few of the things that robots are designed for. Two important aspects of manipulation are gripping (grabbing) and lifting.

Grippers

Grippers are used to pick up and move objects from place to place. There are many types, a few of which are described below.

Linear One-Axis Grip (pinchers)

Imagine grabbing an object, like a soda can, with only your thumb and forefinger. You would probably attempt to grab the can by either 1) pinching it with the tips of your fingers or 2) encircling it with the entire length of the arc formed by your thumb and forefinger. In the same way, a linear one-axis grip holds an object with either the tips of its arms or by encircling it with its arms, the way you would hold onto a very large beach ball. For round objects, the second method requires less force to hold an object, as the greater contact surface provides more friction. The first method also requires that the "finger tips" be positioned very precisely, but is better for picking up small, light objects of irregular shape.



Linear Two-Axis Grip (claws)

Now envision grabbing an object with 3 fingers – this is how a two-axis grip works. The advantage here is that you don't have to pinch the object or hold it solely by friction, as the object can be nestled within the fingers of the claw without being squeezed by them.

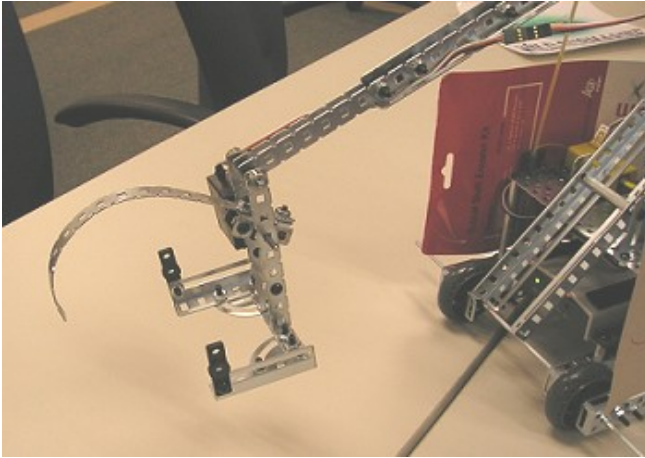


Photo used by permission from Gary Yeap & FVC 3614 ('06-'07)
TBA (Tee-Ba)

Roller Grip

A computer printer with roller feed uses a roller grip – the spinning rollers on either side use friction to move the paper into place. In theory, by reversing the direction of the rollers, you could spit the paper back into the paper tray. Depending on the size and shape of the object, a roller grip can have very good holding power. Roller grips fashioned from tank treads have become a popular staple in Vex competitions (see below). Another example is S.P.A.M.'s robot (pictured in the Chassis section) which uses a conveyor mechanism for gripping balls.

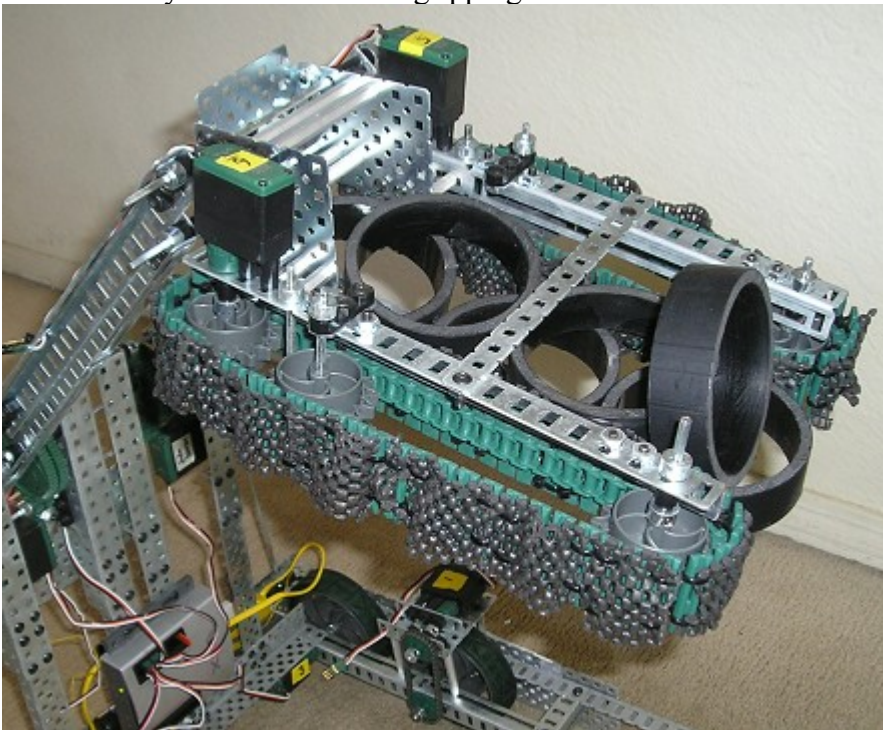
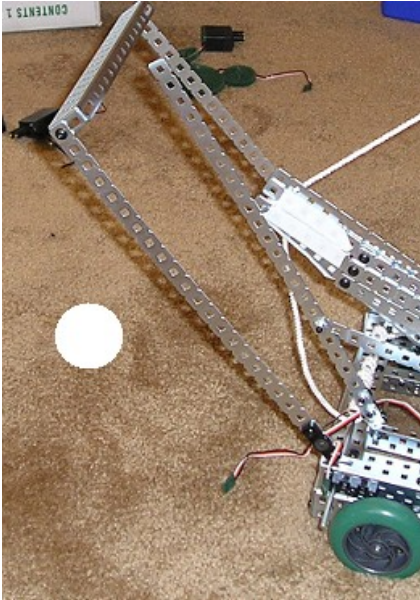


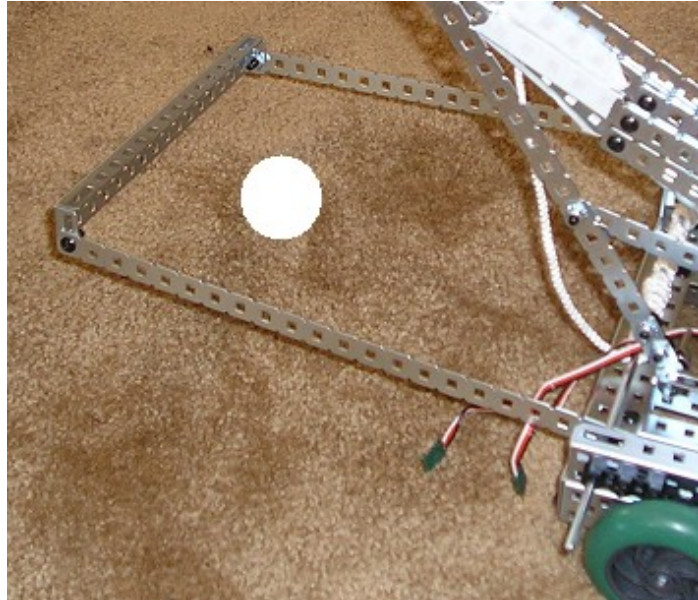
Photo used by permission from FTC 438 ('07-'08) Metal Gear

Hoop Grip

A hoop grip traps objects by surrounding an object with a hoop (covered or uncovered), then releasing it by lifting the hoop (or dragging the object and allowing it to drop to the ground). While simple to construct, it tends to be unreliable and is not generally recommended.



"Ball" released



"Ball" trapped

Choosing an Optimal Gripper

There are a number of factors to consider in designing grippers. First, the size, shape, and weight of the object to be picked up will affect which type of grip is optimal. Whether the object is on or off the ground and where it needs to be moved will also play a role.

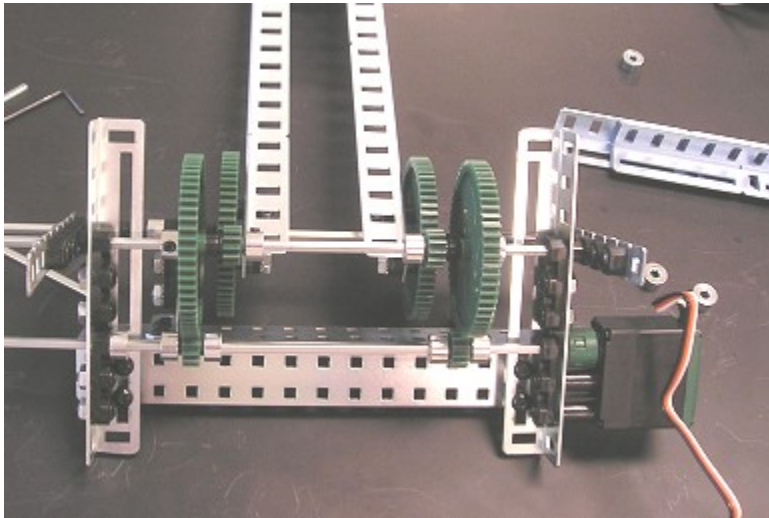
Linear grippers are versatile; that is, they can pick up objects of a variety of sizes and shapes, and carry objects to and from a variety of locations -- objects do not need to be dragged or rolled along the ground. They do require that the arm be precisely placed in relation to the object to be grabbed. Roller grips are much more forgiving in this way, as the entire length of the roller is used for gripping. Hoop grips tend to be the least versatile, as they often require that objects be dragged or dropped, rather than carried.

Speed of grabbing and release are important considerations. A linear grip which is powered by pneumatics will operate more quickly than a motorized grip. A roller grip may be slow for a single object but can grab a large volume of objects more quickly than a linear grip.

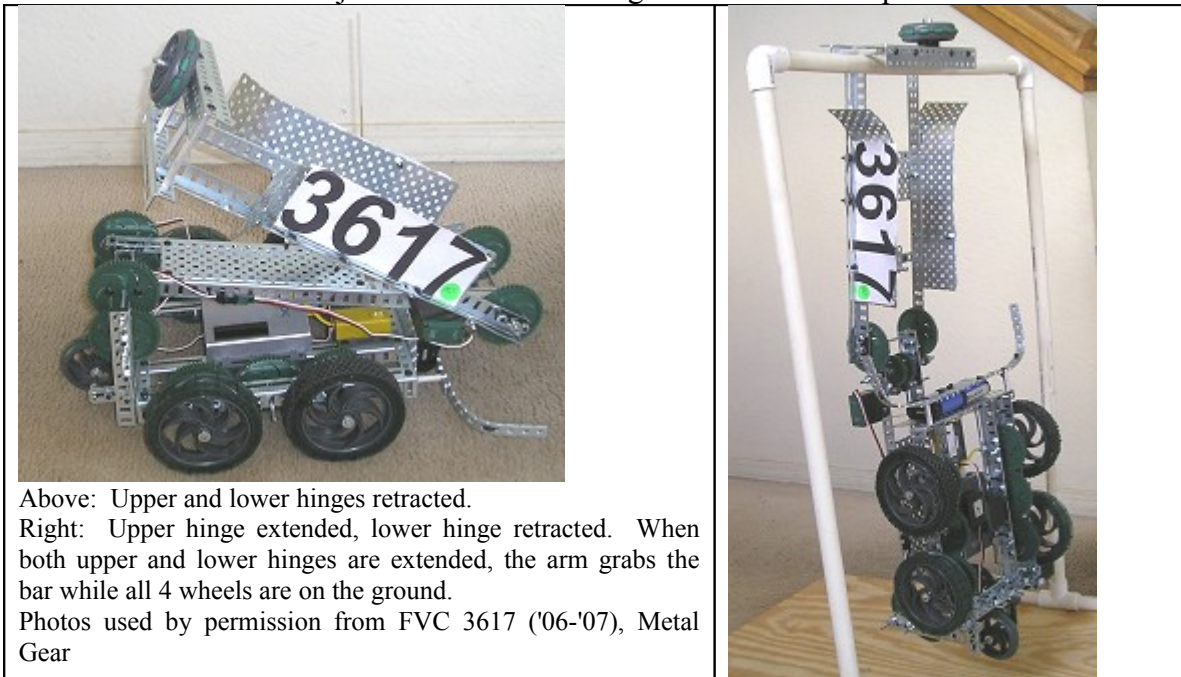
The ability of a gripper to "hold on" is another consideration. A pneumatic grip can hold on without being powered because of its bi-stable (open or closed) configuration. A motorized one-axis grip must be continually powered while carrying an object because continued pressure/friction is required to keep hold of the object. The disadvantages of the pneumatic grip are that the parts are expensive, and use of pneumatics is not permitted in all competitions. Roller grips hold on without continual powering.

Arms

A robot may be required to lift heavy objects (including itself), and there are a number of ways to accomplish this. The simplest device is an arm with a single axle driven by a motor that rotates the arm up and down. Generally, connecting an arm directly to a motor is not desirable, because the motor has too much speed and not enough torque for most practical purposes – gearing down in 1 or 2 stages produces a better design. The arm shown below uses 2-stage reduction, with a 1:25 gear ratio (1:5 and 1:5). It can be attached to the back of a Squarebot or other robot. Note that double gearing (duplicates on right and left) adds strength and reduces the likelihood of stripping the gears.



Additional powered "joints" can be added, which should also be geared down. The robot below has a 2-jointed arm and is designed to chin itself up.



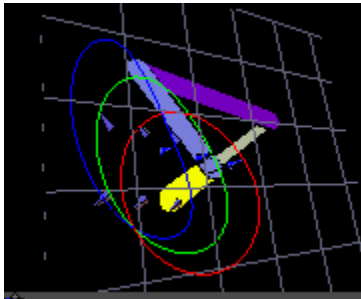
Above: Upper and lower hinges retracted.
Right: Upper hinge extended, lower hinge retracted. When both upper and lower hinges are extended, the arm grabs the bar while all 4 wheels are on the ground.
Photos used by permission from FVC 3617 ('06-'07), Metal Gear

Lifts

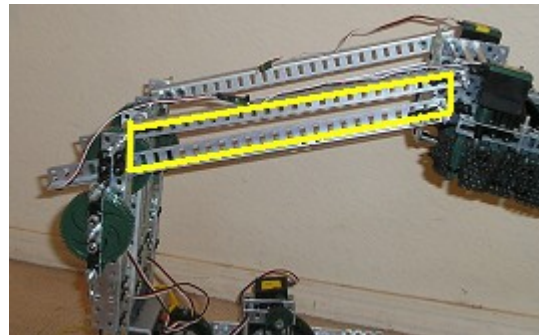
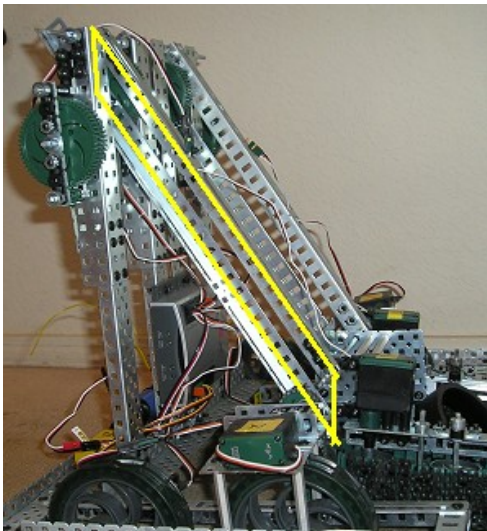
Articulated lifts use multiple jointed sections that unfold to lift a platform or bucket. An example of this is the four-bar linkage, described below. *Telescoping lifts* have multiple overlapping sections that move into and out of one another. Two examples of these are extension lifts and scissor lifts, which will be discussed in more detail.

Four-Bar Linkage

Circular motion produced by motors can be translated into linear lifting motion. The four-bar linkage is one way to accomplish this. Check out the animated gif: www.linkagedesigner.com/res/fourbar.gif⁷



The gray bar is fixed at both ends, and as the yellow bar travels in a circular pattern, the end of the purple bar (where it is joined to the blue bar), oscillates up and down. Some practical examples of this principle are the windshield wipers of a car and a crank and piston. Below is a Vex™ example of a four-bar lift. This design was chosen because it allows the manipulator on the front to remain (more or less) parallel to the ground.

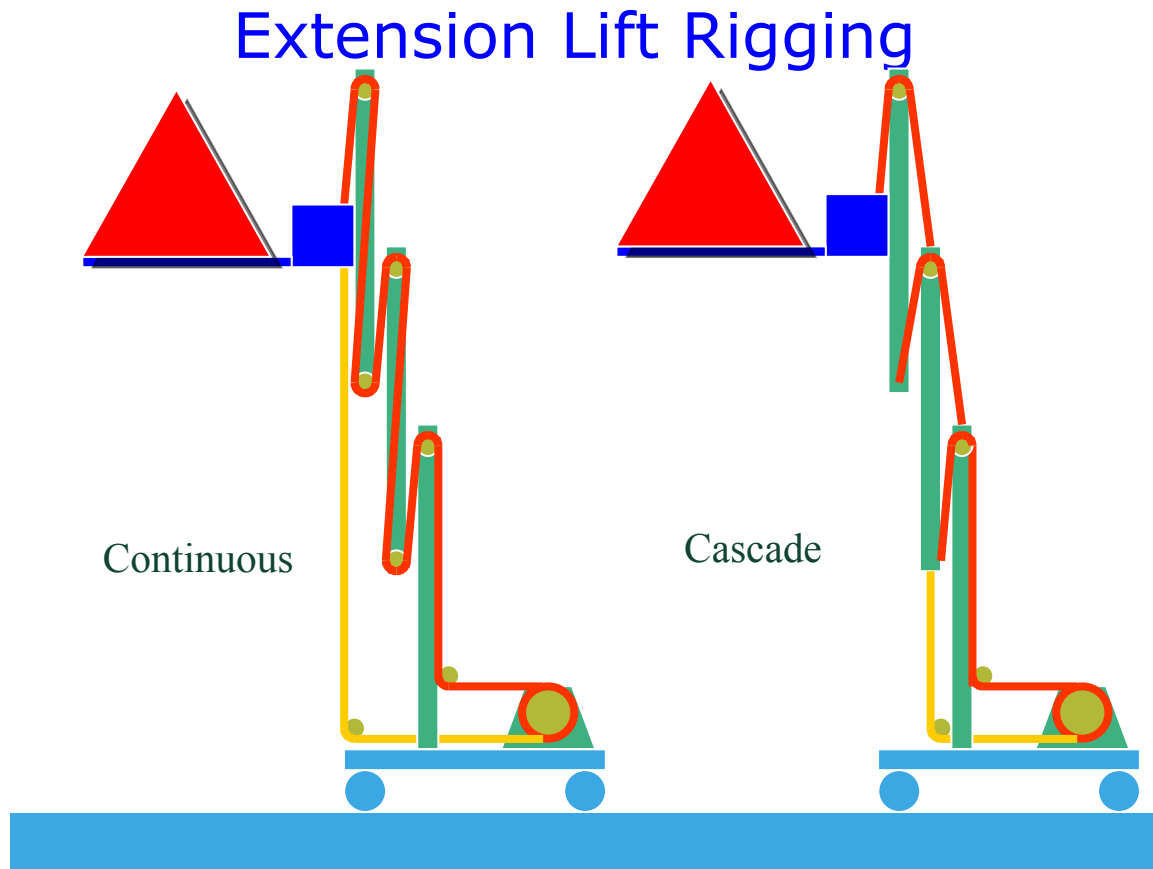


Photos used by permission from FTC 438 ('07-'08)
Metal Gear

Another example of a 4-bar arm (in action) is the BottleBot at: vexlabs.com/vex-bottlebot.shtml.

Extension Lifts

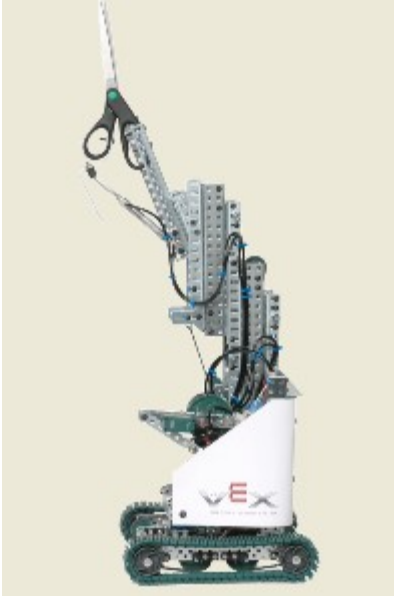
An extension lift also allows an arm to lengthen in a linear fashion by tightening a cable or chain around a drum, causing overlapping sections to extend or retract. Powering is required for both extension (going up) and retraction (going down). Greg Needel's PowerPoint⁵ contains the diagram shown below:



Each design has its own advantages and disadvantages. Lifts with *continuous rigging* move evenly but slowly, as more cable is required for full extension. The cable uses identical speeds for going up and going down, so only one drum is required. The disadvantages are that the intermediate sections tend to be more prone to jamming, due to the longer cable length and lower cable tension. In addition, the longer cable requires that more cable be moved in the same amount of time, requiring a faster motor.

Lifts with *cascade rigging* require that less cable be moved to fully extend, and the intermediate sections are less likely to jam. However, the cables for going up and going down use different drums requiring different speeds, and higher tension on the lower-stage cables requires higher gearing to deal with the greater force. The greater number of connections increases the complexity of the system and potential for something to go wrong.

The ribbon cutter robot below by John V-Neun and Chris Carnevale uses a 5-stage extension lift. You can view more detailed photos and a video of its operation at: vexlabs.com/vex-ribbon-cutter.shtml.



Retracted



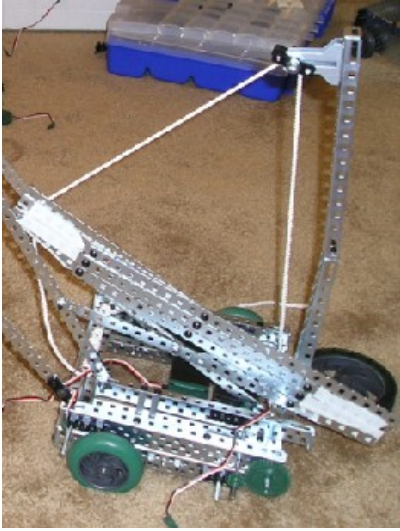
Extended

Photos used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI.

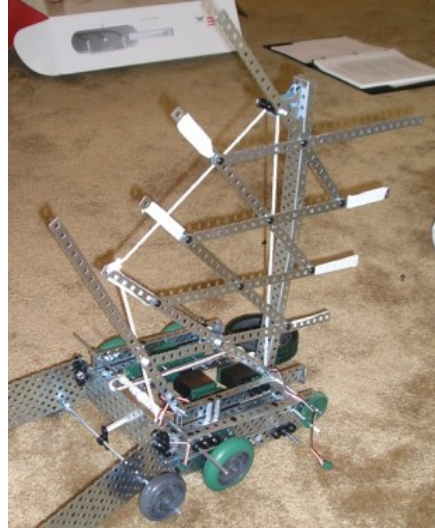
Scissor Lifts

Scissor lifts also have overlapping sections that extend in a criss-cross pattern. One advantage that they have over extension lifts is that the retracted height is minimal, enabling the lift to easily get under a load. The disadvantages are that the large number of criss-cross bars increases the weight, and that the stability decreases as height increases and the lift approaches maximum extension.

The lift below is activated by a pulley. The taped ends were eventually cut off when the desired length was found by trial and error – measure twice, cut once. The use of a single thickness of long bars was a poor design choice, as the long bars tended to bend when the robot attempted to lift itself. A better choice would have been to use a double thickness in the scissor lift, separated by spacers or short threaded beams. Although this would add to the weight of the arm, the increase in strength and stability would more than compensate.



Retracted



Extended

Programming

Currently, the 3 programming languages that are marketed for the Vex controller are EasyC 2.0 (drag & drop commands, menu driven), ROBOTC (text-based, menu driven), and MPLAB IDE (for advanced users). For a detailed comparison, see: <http://www.vexrobotics.com/vex-robotics-programming-kit.shtml>. Because this is a beginner's guide, the documentation here is for EasyC. However, a free trial version of ROBOTC and sample lessons from the tutorial can be found at: http://www.robotc.net/content/vex_curric/vex_curric.html.

The tutorial which comes with the EasyC programming kit (printed pages to be inserted into the Inventor's Guide manual) provides excellent step-by-step instruction, and it is advisable to work through it without skipping steps. If your programming kit came without the tutorial, it can be downloaded here:

<http://www.vexrobotics.com/docs/inventors-guide/programming-guide.pdf>

The following information supplements the tutorial and presumes that you have completed it. While it is possible to use EasyC for a first programming experience, the tutorial does not teach generic programming concepts in a comprehensive way, and it would be helpful to have mastered important concepts such as inputs, outputs, looping, and conditionals in another introductory programming course.

Remote and Autonomous Control

You can program your robot to operate by remote control as described in the tutorial by using the commands under RC Control on the menu. You can also program your robot to operate completely autonomously, with no control from the transmitter. However, certain versions of EasyC for Vex contain a glitch that requires that prevents autonomous programs from executing unless the microcontroller receives a signal from a transmitter, even if the robot is not affected by any input from the joysticks or buttons on the transmitter. If your robot refuses to function after a program is downloaded, try connecting an RF receiver (yellow box) and turning on a transmitter.

Downloading Tips

Downloading can be a slow process, but it is highly recommended that you exercise patience and allow the process to complete itself. Occasionally, interrupting the process has resulted in a completely frozen and unusable system: not being able to run the old or the new programs, and not being able download anything new (ouch!). If you do experience a system freeze, try removing all batteries and power for a period of time and try again. Sometimes downloading the Master Code (available under Terminal Window Options) will clean out the bugs, but it will also erase your program.

Be aware that your program will begin running the instant that downloading is complete, which may result in your robot making its way to Alaska before you have time to unplug the cable. One solution to this dilemma is to prop up the robot chassis on a platform (books or 2" X 4" blocks work), so the wheels are off the ground while downloading. Another is to use the "interrupt" button, located on the box on the orange download cable.

Motors

According to the manual, motor speeds are regulated by setting the motor direction, with 255 producing the maximum clockwise (CW) speed, 0 producing the maximum counter-clockwise speed, and 127 producing a motor stop. More precisely, the motor speeds increase gradually over the following ranges:

CW rotation: 128 – 255 (128 produces a motor stop, 255 produces the greatest speed)

CCW rotation: 127 – 0 (127 produces a motor stop, 0 produces the greatest speed)

Many simple robot designs will (attempt to) drive straight by using the technique of turning motors on either side at the same speed in opposite directions. To program motors to do this, choose motor directions which add up to 255. For example, if a CW motor is set at 190, the CCW motor should be $255 - 190 = 65$.

The Online Window – A Diagnostic Tool

Actually, a stopped motor can be produced by a range of values, rather than a single value. To find the true range of stop values, use the Online Window, described on page 8.21 in the Programming section. Drag the slider bar through the range of values until you observe your wheels actually moving. A free-spinning axle might begin to move CW at 135, while a loaded motor might require a value of 160 to produce any CW rotation, since it has more friction to overcome. This is an important consideration when trying to program your robot to move slowly. A robot which is programmed too close to the actual stop threshold tends to stall with conditions which add friction, such as adding weight, turning, and driving on different surfaces. In general, setting motors at full power is desirable, unless there is reason to do otherwise. Speed can be reduced mechanically by gearing down.

Multiple Programs

If you want to switch from one program to another without downloading each time, multiple programs can be embedded into a larger program and selected using jumpers (the small orange clips) placed into the digital inputs.

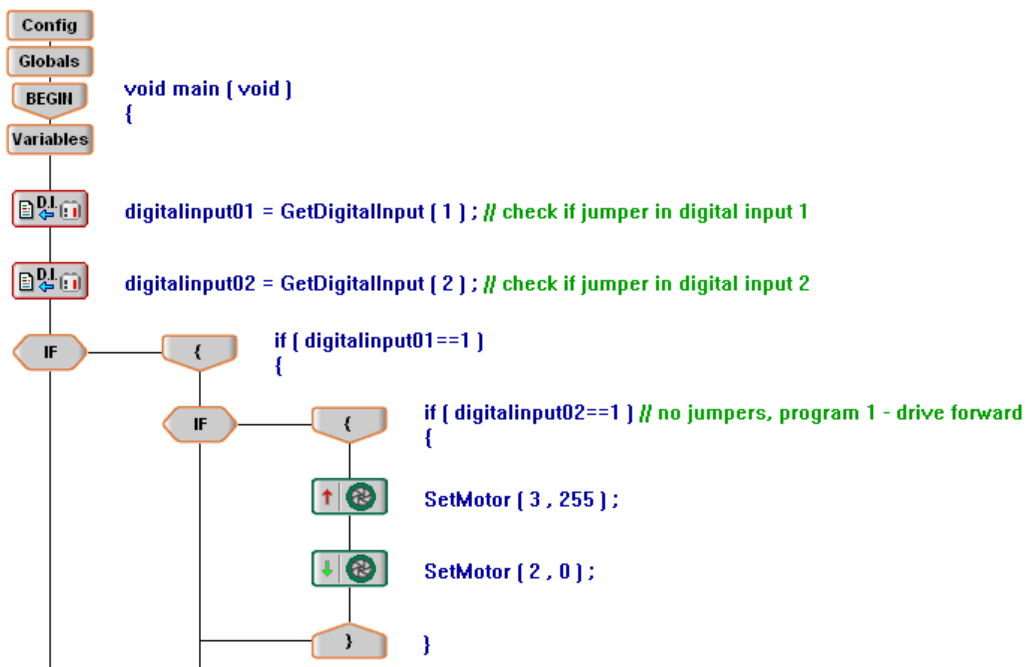
You can test the effect of jumpers on the digital inputs by using the Online Window. The window shows that ports 5 - 10 are configured as digital inputs. On the controller, there is a strip of slots labeled ANALOG/DIGITAL, number from 1 – 16, TX, and RX. Try inserting the jumper clip into the various ports, and note that for digital inputs (ports 5 - 10), the default configuration is "1"; placing a jumper in a digital input slot changes its value to "0."

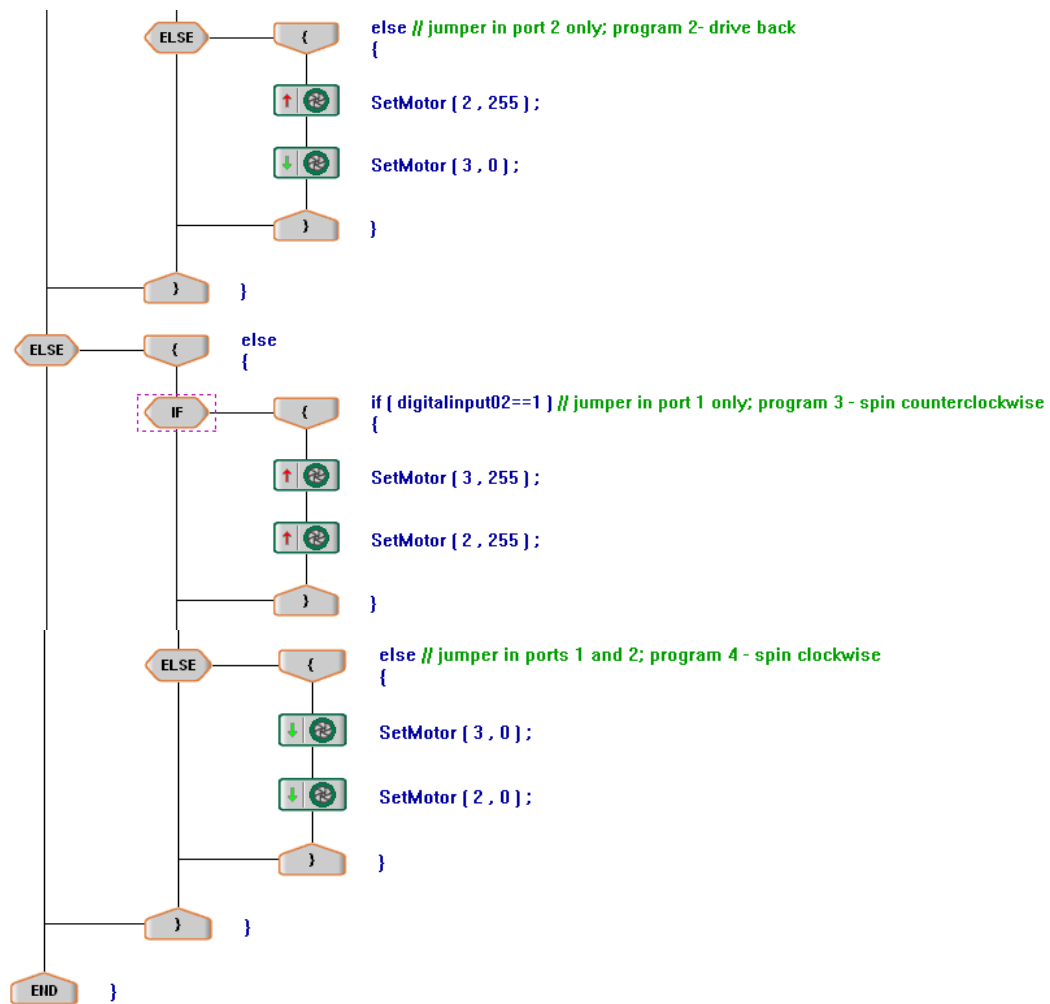
One ergonomic modification is suggested here. Our personal preference is to use ports 1 – 4 for jumpers, because they are on the very end of the port strip – ports in the middle are sometimes hard to see, especially if your controller is boxed in by other parts. If you are not planning to use analog inputs (or at least not the first few), you can reconfigure ports 1-4 as digital inputs. To do this, when you first open your program, double click on the gray "Config" button at the beginning of the program. You should see the layout of all the microcontroller ports. Right click on ports 1 – 4 to toggle between analog input and digital input.

By reading the digital inputs at the beginning of the program, you can select up to 2^n different programs, where n is the number of ports used (think binary). For example, using 3 ports gives $2^3 = 8$ options in the following configuration:

Program	Port 1	Port 2	Port 3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Below is a program that uses 2 digital inputs (inputs 1 and 2 are reconfigured as digital) for a total of 4 subprograms. To activate each program separately, insert the jumper(s) which will select the desired program before turning on the robot. The selected program will continue executing until the robot is turned off. You cannot switch from one program to another by switching the clips while the robot is running, unless you write your program in such a way that it continuously polls for the digital inputs in an infinite loop.





```

1 #include "Main.h"
2
3 void main ( void )
4 {
5     int digitalinput01;
6     int digitalinput02;
7
8     digitalinput01 = GetDigitalInput ( 1 ) ; // check if jumper in digital input 1
9     digitalinput02 = GetDigitalInput ( 2 ) ; // check if jumper in digital input 2
10    if ( digitalinput01==1 )
11    {
12        if ( digitalinput02==1 ) // no jumpers, program 1 - drive forward
13        {
14            SetMotor ( 3 , 255 ) ;
15            SetMotor ( 2 , 0 ) ;
16        }
17        else // jumper in port 2 only; program 2- drive back
18        {
19            SetMotor ( 2 , 255 ) ;
20            SetMotor ( 3 , 0 ) ;
21        }
22    }
23    else
24    {
25        if ( digitalinput02==1 ) // jumper in port 1 only; program 3 - spin counterclockwise
26        {
27            SetMotor ( 3 , 255 ) ;
28            SetMotor ( 2 , 255 ) ;
29        }
30        else // jumper in ports 1 and 2; program 4 - spin clockwise
31        {
32            SetMotor ( 3 , 0 ) ;
33            SetMotor ( 2 , 0 ) ;
34        }
35    }
36 }

```

Sensors

A robot can be programmed to go from one point to another simply by instructing the motors to turn on and off at the appropriate times, causing it to turn or go straight. Doing this is akin to attempting to drive to the store, blindfolded and with earplugs, hoping to stay on course by knowing the appropriate turns and distances. In real life, a driver is constantly monitoring the environment, listening for the sound of oncoming traffic, looking for obstacles, and braking when he feels the tires hit a bump. Adding sensors allows your robot to monitor and react to the environment in this way.

The programming software comes with test programs which demonstrate how to get sensor readings and show you the readings of the sensors onscreen. To access the programs, click File > Open Project > Test Code. From there, select the sensor you want to test. Specific examples of robots programmed to use sensors can be found in *Vex™ Machinations: A Step-By-Step Project Guide*.

Again, the Online Window is an excellent diagnostic tool, as it shows the readings on all sensor inputs when activated.

Bumper Sensor & Limit Sensor

The use of the bumper sensor to control the actions of the robot is described in detail in the tutorial that comes with the programming kit ("program three: using motors and sensors together," and "program four: reversing and turning," beginning from p. 8.38). The limit sensor is also digital (its only readings are 1 and 0, where 0 = pressed, 1 = released), which is why it is designated as a switch.

One of the disadvantages of using the bumper or limit switch alone to detect whether a robot has hit an obstacle is that the sensor itself must contact the obstacle. If another part, like a corner, of the robot bumps into something, the robot will continue to move forward. This problem can be rectified by "fronting" your robot with a wider bumper, which activates the switch when any part of the attachment comes into contact with an obstacle.

Line Follower

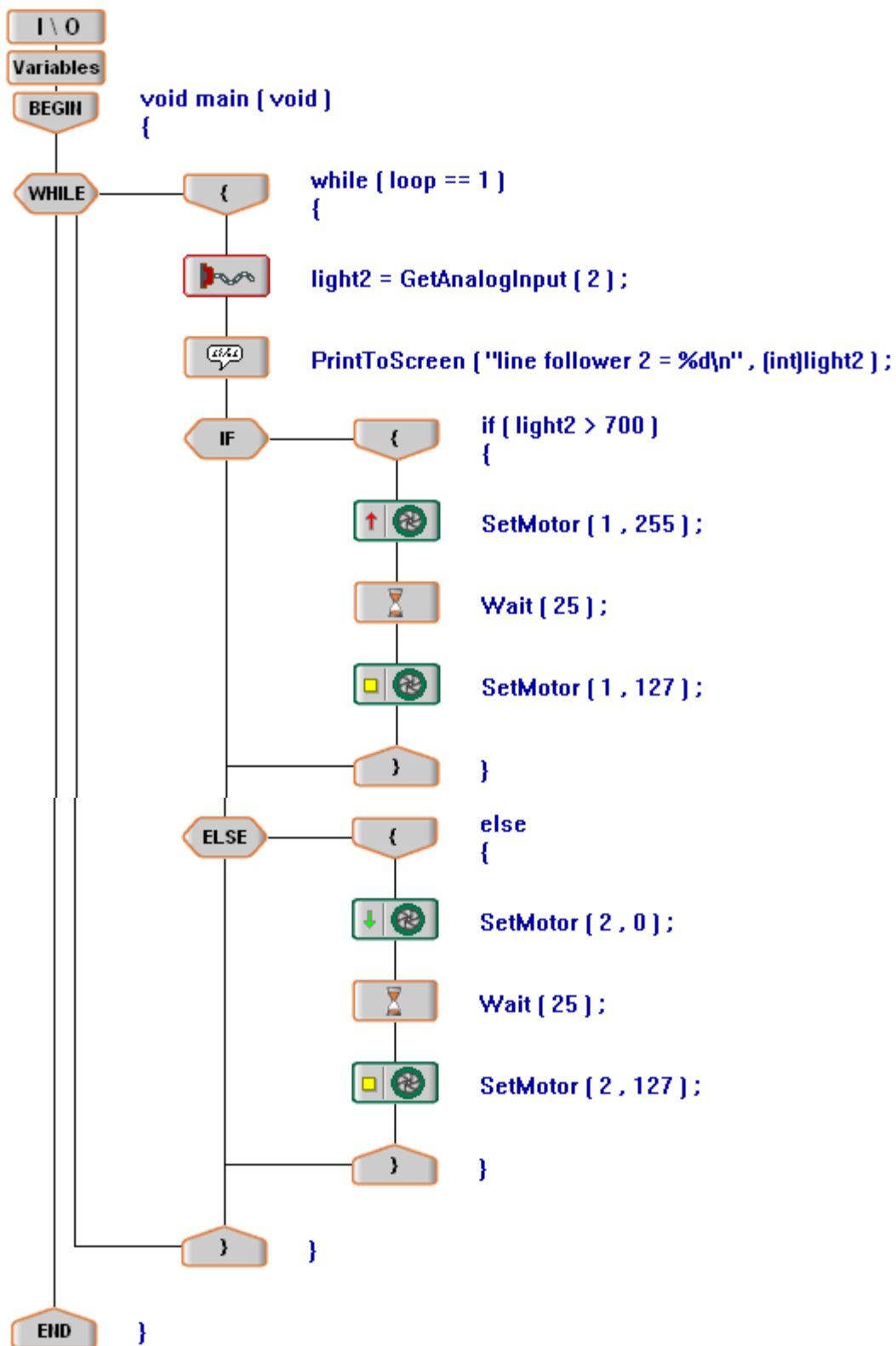
The line tracker kit contains 3 sensors, but you don't need to use all three of them at once to follow a line. Suppose you want to follow a black oval ring on a white mat. You can use the LINEFOLLOWERTEST program in the Test Code folder to obtain sensor readings when the sensor rests on the black and on the white. These readings will fluctuate, depending on how close the sensor is to the mat. However, it should be clear which ranges of values indicate "dark" and "light," and a suitable threshold set to distinguish one from the other.

A one-sensor line follower actually follows the edge between black and white, rather than the line itself. It begins on one side of the line, say the right, and "looks for" the line on the left. When sensor readings cross the threshold from light to dark, the robot turns away from the line briefly, then turns toward it again, until it finds it. It continues to follow the edge between white and black, as long as the black is on the left and the white is on the right. The robot can also be programmed to follow the line starting from the left side.

Sometimes a line follower will spin in circles because it is looking for a line, say on the left, and the line isn't there. This indicates that the sensor has failed to detect the line; most likely the robot has driven over it without detecting it. This happens when the robot has driven too quickly or the sensor has "looked for" the line too infrequently. The problem can usually be rectified by slowing down the robot or decreasing the interval between sensor readings. Because of the overshoot dilemma, wide lines are easier to follow than narrow ones, and gentle turns are easier to follow than sharp corners.

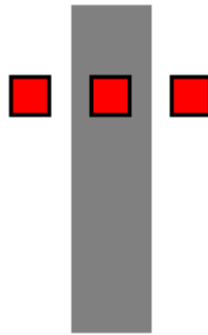
The program below uses one sensor to follow a line on its right side. It uses the Slowerbot (see Locomotion section, 4-wheel drive) rather than the Squarebot to reduce the overshoot. A threshold value of 700 was chosen, since "light" readings ranged between 30 and 650, and "dark" readings were in the 800+ range.

```
1 #include "UserAPI.h"
2
3 int light2;
4 int loop = 1;
5
6 void main ( void )
7 {
8     while ( loop == 1 )
9     {
10         light2 = GetAnalogInput ( 2 ) ;
11         PrintToScreen ( "line follower 2 = %d\n" , (int)light2 )
12         if ( light2 > 700 )
13         {
14             SetMotor ( 1 , 255 ) ;
15             Wait ( 25 ) ;
16             SetMotor ( 1 , 127 ) ;
17         }
18         else
19         {
20             SetMotor ( 2 , 0 ) ;
21             Wait ( 25 ) ;
22             SetMotor ( 2 , 127 ) ;
23         }
24     }
25 }
```



It should be noted that greater speed and precision can be obtained by using two or three sensors for line following. In a two-sensor configuration, the sensors straddle the lines. The left sensor "looks for" the line on its right and turns away to the left when it finds it; then, the right sensor "looks for" the line on its left and turns away when it finds it. When neither sensor "sees" black, the robot drives straight by default. Wider spacing between sensors allows the line follower to go faster, since it can continue in a straight direction until it hits the other sensor. Narrower spacing results in a slower robot, but one which follows the line with less side-to-side variation.

It is also possible to use three sensors⁸, an arrangement commonly found commercially. The sensors are spaced so that at least one sensor will detect the line at any given time. If the left sensor sees black, the robot turns left until the center sensor sees black, and if the right sensor sees black, it turns right until the center sensor sees black. When the center sensor sees black, the robot goes straight by default.



Optical Shaft Encoder (rotation sensor)

The optical shaft encoder records the number of rotations that the axle turns in the shaft, where a reading of 90 = one complete rotation. This, in turn, can be used to calculate the distance that a robot has traveled by multiplying by the circumference of the wheel.

$$\text{distance traveled} = \frac{\text{sensor_value}}{90} \times 2 \cdot \pi \cdot r$$

One unfortunate aspect of the shaft encoder is that it does not use positive and negative numbers to record clockwise vs. counterclockwise rotation. Thus, a reading of 90 might indicate a full CW rotation, a full CCW rotation, or a half-rotation CW followed by a half-rotation CCW. Thus, in your programming, it is very important to store the readings in a variable every time the direction of motion changes and reset the encoder when necessary.

Ultrasonic Sensor

The ultrasonic sensor detects distance by sending out a signal and recording the time it takes for the reflected signal to return. The number returned indicates the number of inches the robot is from a barrier. Moving robots produce significant fluctuations, as the distance changes from the time the signal is sent until the time it is received. An artifact of 99 (inches) periodically pops into the readings, especially if the robot is moving or changing direction, as the receiver cannot lock onto the signal. Where

necessary, programming can be used to filter out these readings. The documentation that comes with the ultrasonic sensor also describes how other factors, such as temperature and altitude, can affect the speed of sound. These will play a role in the readings obtained.

Light Sensor

The light sensor differs from the line tracker, in that the light sensor is designed to read the intensity of visible light, while the line tracking sensors both emit and receive infrared light. Traditionally, the most common application of light sensors has been line-following, but a light sensor can also be used to detect light in a dark room, as well as distinguish between certain colors (which are read as darker or lighter shades of gray).

Transmitter Considerations

Crystal Upgrade

Unfortunately, most Vex™ Starter Kits come with the same frequency module & crystal (#61), which means that signals from 2 remote-driven robots will interfere with each other, causing jerky, erratic motion. In fact, if you wish to use 2 transmitters to control different motors on the same robot, you will encounter the same problem. One solution is to purchase the Crystal Upgrade kit, which has sets of crystals and frequency modules tuned to different frequencies than the commonly used ones. Once replaced, this should prevent your robot from receiving the transmissions of other teams (unless by Murphy's Law, they have purchased crystals of the same frequency as you). This is a possibility, as there are only 2 Crystal Upgrade kits with a total of 8 different frequencies available at the time of this writing. WiFi inserts are in development and hopefully will be released soon, which should solve the crystal dilemma.

Tethering

One way to completely ensure that your transmitter will not interfere with another is to bypass the RF receiver(s) by transmitting signals to the robot via a tether. Using a phone cord (the curly cord, not the straight one going to the wall), connect the tether port on the back of the transmitter directly to one of the Rx ports on the controller. Since there are 2 Rx ports, you can connect 2 transmitters in this way. The downside is that you have to follow your robot around with the transmitter(s) and watch for tangling of cords. This is adequate for practice, but can't be used in most competitions. However, if you plan to attend a competition, it's advisable to bring a tether cord for use in the practice area, so that your robot's signals don't interfere with others and vice versa.

Programming the Transmitter (rather than the microcontroller)

Your transmitter can be programmed manually (without a computer) so that the joysticks and buttons operate differently from the default, as described in the *Inventor's Guide*, Appendix E – Control Configurations. However, reprogramming the transmitter should be done with caution, especially if you own more than one transmitter. Realize that switching from one transmitter to another (differently programmed) will cause your robot to operate differently when you activate the same controls; for example the robot might move forward when you push both joysticks forward on one transmitter, but spin when you push both joysticks forward using a different transmitter. Rather than having to learn new controls for every transmitter, you can save the driver some confusion by carefully labeling the transmitters or reprogramming every transmitter in the same way.

Alternatively, if you want the controls to operate in a certain way, you can modify the configuration through the software that you download. This way, all transmitters can be left in “default” mode and will produce identical results, preventing unwanted surprises.

Calibration

Occasionally, plugged-in motors will move on their own (without pushing the joysticks or channel buttons), and the joysticks or buttons must be pushed for them to stop. You can re-calibrate your transmitter to “zero” at the default position by selecting

the channel ("Select Channel" button on transmitter front) and pressing the "Data Input +/-" button until the motor stops at the resting position.

Simple Challenges

You can experiment with different robot designs by tackling some simple challenges which require little or no field construction. These are patterned after the Vex™ Pentathlon events of the Robofest competition (www.robofest.net). While they are optimal for two or more competing robots, one-robot challenges are possible.

Tug of War

Attach a vertical partially threaded beam (for a "hitching post") to the chassis of each robot. Loop a rope around each hitching post, and see which robot drags the other across a line. For one robot, use a bungee cord, and see if you can increase the amount of stretch by modifying your robot. Alternatively, drag a weighted platform on the ground, and see if you can modify your robot to drag increasing amounts of weight

Speed Race

Line up the robots at the starting block and let 'em rip. For one robot, time yourself for personal best. For an added challenge, use black electrical tape for the starting and finish lines, and require robots to start and stop at the appointed markers.

Obstacle Course

Set up a maze or obstacle course using 2-liter bottles or soda cans. Maximum speed is desirable, with penalties for obstacles touched, moved, or knocked over.

Race Track

Make an oval race track (butcher paper with black electrical tape works), and design the fastest line follower possible to make its way around the track.

Skee Ball

Lob a maximum number of ping-pong balls into a desired target (like a waste basket) in a fixed amount of time (or a set number of trials). For a greater challenge, mount rectangular wastebaskets onto the rungs of a ladder, and assign points for each level.

Further Exploration

Innovation First, Inc. (IFI)

IFI's website, vexrobotics.com has photos and video clips of several interesting designs that demonstrate the extent of how much you can do with the Vex™ system. Check out the video clips at vexlabs.com/vex-robot-photos.shtml, and follow the links >Vex robotics – Index of Galleries > Robots > Ribbon Cutter Robot (additional robots can be found here as well). IFI is currently the distributor of Vex™ components, so they have a vested interest in providing ongoing support.

Vex Forum

The IFI Official Community Site is located at vexforum.com. The Community section has a Robot Showcase with many examples of Vex robots. Some sections are "official" and posted questions can only be answered by experts. Other areas are interactive, and questions here usually receive prompt responses. Use the "Search" function before posting questions, as your question may have already been asked and answered.

Chief Delphi Forum

Follow the thread chiefdelphi.com/forums/forumdisplay.php?f=162 to the Vex Challenge subforum on Chief Delphi and click on "Show Threads", selecting "From the Beginning." There is a wealth of information in old posts, as this forum had its greatest level of activity from 2005-2007, when Vex was the system used by FTC.

Robot Magazine

Every now and then, Robot Magazine (botmag.com), will have articles related to Vex™. Here's one:

botmag.com/articles/mythbusters_test_the_vex_robotics_design_system_1.shtml

User Dedicated Websites

Dozens of Vex aficionados have websites dedicated to Vex. Here are a few good ones:

- <http://www.tonybuildsrobots.tk/>
- <http://ducttape477.vexrobotics.googlepages.com>
- <http://www.vexplosion.com/Home.html> (still in development, but a good collection of resources)

Search Engines

Try searching YouTube and Googling "Vex Robots" and similar entries.

References

- ¹Innovation First, Inc. (IFI) vexlabs.com
- ²Chief Delphi forum (Vex subforum) chiefdelphi.com/forums/forumdisplay.php?f=146.
Try the search function before posting a question!
- ³FIRSTwiki: http://wiki.chiefdelphi.com/index.php/Main_Page. FIRSTwiki has some good overviews of engineering concepts, but the coverage is spotty. Also, the information is geared more toward FRC (FIRST Robotics Competition) robots, rather than Vex™, though many of the concepts are transferable.
- ⁴Differential (mechanical device). (2008, April 28). In *Wikipedia, The Free Encyclopedia*. Retrieved 00:47, May 1, 2008, from http://en.wikipedia.org/w/index.php?title=Differential_%28mechanical_device%29&oldid=208699551
- ⁵Pilvines, John, and Team Unlimited (FVC 2013). FLL to FVC – a Case Study, Power Point Presentation, *FIRST* Robotics Conference Workshop, April 2007. You can download this presentation at: <http://eaglevex.syraweb.org>.
- ⁶Needel, Greg. Building Competitive Manipulators: The Mechanics and Strategy, Power Point Presentation, *FIRST* Robotics Conference Workshop, April 2006. Although this presentation was given by Greg Needel in 2006, he credits Andy Baker for much of the material, who credits Chris Husmann. You can download this presentation at: www.usfirst.org/robotics/2006/Workshops/wrkpresentations.htm.
- ⁷LinkageDesigner. www.linkagedesigner.com/res/fourbar.gif
- ⁸Dubel, William. Reliable Line Tracking. www.dubel.org

Appendix A: Recommendations for Add-ons

(Prices from May 2008 subject to change)

Opinions on most valuable parts will vary greatly between users. There are a number of ways to upgrade your Starter Kit:

- Purchase items on the "Bang for Buck" upgrade list to greatly increase the functionality of your kit for just over \$100. Gradually add other components, such as the Power Pack & Programming Kit, then mechanical upgrades and sensors as needed. Competition teams often do not purchase additional sensors, as many Vex competitions so far have been mechanics-heavy.
- Purchase a second Vex Starter Kit. This is a cost-effective way to add extra motors, wheels, transmitter & receiver, metal, and hardware, items which are commonly used.
- The Metal and Hardware Kit (\$79.99) and Tri-Pack Bundle (\$149.99) composed of the Advanced Metal Pack, Advanced Motion Pack, and Advanced Hardware Pack also add structural items at a discount. Some items may not be commonly used, and adding only commonly used pieces may be more economical.
- If you haven't purchased a Starter Kit yet, consider the Classroom Lab Kit at \$549. It contains most of the items on the Bang for Buck list, other mechanical upgrades, and more. Purchase a Programming Kit separately for \$99.
- The Educational Link also contains Lab Expansion packs: a Sensor Pack (Level 2 upgrade, \$99) and Advanced Drive System and Mechanisms Pack (Level 3 upgrade, \$249). These are both a good value. The Pneumatics Starter Kit (Level 4 upgrade \$289) is highly specialized – if you don't know that you'll need it, you probably won't.

The Bang for Buck Upgrade List (in order of priority)					
Part Name	Part #	Where on Website	#	Price	Total
Gear Kit	P/N: 276-2169	Vex Standard Parts	1	\$12.99	\$12.99
Drive Shaft Square Bar Pack	SQR-BAR-120-2PK	Vex Starter Kit Parts	2	\$4.49	\$8.98
Motor Kit	P/N: 276-2163	Vex Standard Parts	1	\$19.99	\$19.99
Collar Pack	COLLAR-16PK	Vex Starter Kit Parts	2	\$10.49	\$20.98
Angle 1 X 1 , Inverse	P/N: ANGLE-001R-4PK	Vex Metal Parts	1	\$14.99	\$14.99
1/2 in. 8-32 Screw Pack	SCREW-832-12-28PK	Vex Starter Kit Parts	2	\$1.49	\$2.98
3/4 in. 8-32 Screw Pack	SCREW-832-34-14PK	Vex Starter Kit Parts	2	\$0.99	\$1.98
Delrin Bearing 10-pack	BEARING-FLAT-10PK	Vex Starter Kit Parts	1	\$4.99	\$4.99
Keps Nuts 8-32 65-pack	KEPS-NUT-832-65PK	Vex Fasteners	1	\$1.99	\$1.99
6-32 Stainless Steel 1/2 Screw 50 pack	SCREW-632-0250-50PK	Vex Fasteners	1	\$4.95	\$4.95
Open End/Allen Wrench Pack	WRENCH-ALLEN-TOOL-PK	Vex Starter Kit Parts	1	\$2.99	\$2.99
15 X 5 Hole Plate	PLATE-5-15-2PK	Vex Starter Kit Parts	1	\$4.49	\$4.49
Bar 1 X 25 Holes pack	BAR-25-4PK	Vex Starter Kit Parts	1	\$5.49	\$5.49
Total					\$107.79

Ubiquitous Favorites (needed for many competitions)					
Vex Power Pack	P/N: 230-0036	Vex Standard Parts	1	\$49.99	\$49.99
Programming Kit (EasyC/ROBOTC/MPLAB)	P/N: 276-2152 P/N: 276-2188/?	Vex Standard Parts	1	\$99.99	\$99.99

Commonly Used Kits for Complex Mechanics (recommended in order of priority)					
Wheel Kit	P/N: 276-2164	Vex Standard Parts	1	\$29.99	\$29.99
Tank Tread Kit	P/N: 276-2168	Vex Standard Parts	1	\$29.99	\$29.99
Omni Directional Wheel (set of 2)	P/N: 276-2165	Vex Standard Parts	1	\$19.99	\$19.99
Chain and Sprocket Kit	P/N: 276-2166	Vex Standard Parts	1	\$29.99	\$29.99
Advanced Gear Kit	P/N: 276-2184	Vex Standard Parts	1	\$19.99	\$19.99
PWM Bundle	P/N: CABLE-PWM- BUNDLE	Cables and Other	1	\$39.99	\$39.99
Motor Kit	P/N: 276-2163	Vex Standard Parts	2	\$19.99	\$39.98
Advanced Metal Pack	P/N: ADV-METAL-PK	Vex Metal Parts	1	\$69.99	\$69.99
Pop Rivets	P/N: 276-2215	Vex Fasteners	1	\$7.99	\$7.99
Large Omni Wheel (set of 2)	?? (link broken)	Vex Standard Parts	1	\$24.99	\$24.99
Total					\$310.89

Recommended Sensors (in order of priority)					
Line Tracking Kit	P/N: 276-2154	Vex Standard Parts	1	\$39.99	\$39.99
Optical Shaft Encoder	P/N: 276-2156	Vex Standard Parts	1	\$19.99	\$19.99
Ultrasonic Range Finder	P/N: 276-2155	Vex Standard Parts	1	\$29.99	\$29.99
Light Sensor Kit	P/N: 276-2158	Vex Standard Parts	1	\$19.99	\$19.99
Total					\$109.96

Note: Bumper and Limit Switches are included in the Starter kit; additional purchases not recommended.

There are many other items that are useful for specific purposes. These include (but are not limited to):

- Crystal Upgrade - Necessary if you plan to use 2 or more transmitters at the same time, especially good for teams with multiple robots
- Chassis Kits – Useful for bigger chasses, but you can put together your own with enough metal.
- Pneumatics Kits - Cool, but costly
- Aluminum Kits – For lighter robots

Appendix B: Cutting Suggestions for Commonly Used Pieces

Recommended if you purchase the Bang for Buck items listed in Appendix A (these items appear in # added).

Part name	# in starter kit	# added	total #	# cut	# uncut
Angle 1 X 1	4	0	4	2	2
Angle 1 X 1, Inverse	0	4	4	2	2
Drive Shaft Square Bar	2	4	6	4	2
Plate 25 X 5	0	6	6	4	2
Bar 1 X 25	4	2	6	4	2

How to cut

1. Angles 1 X 1 (15" long) – 4 total
Cut one piece into 10" + 5" (on the notch)
Cut one piece into 12.5" + 2.5" (on the notch)
Leave 2 pieces uncut
2. Angles 1 X 1, inverse (15" long) – 4 total
Cut one piece into 10" + 5" (on the notch)
Cut one piece into 12.5" + 2.5" (on the notch)
Leave 2 pieces uncut
3. Drive Shaft Square Bars (12" long) – 6 total (no notches)
Cut one piece into 6" + 6"
Cut 2 pieces into 8" + 4"
Cut one piece into 4" + 4" + 4"
Leave 2 pieces uncut
4. Long bars and plates
It is recommended that you don't pre-cut the thin long bars and plates. These are easy to cut on the spot using tin snips, and they are less conducive to having frequently used, standardized lengths.